In Proceedings of the 1996 ACM SIGMOD Workshop on Data Mining and Knowledge Discovery

Security and Privacy Implications of Data Mining

Chris Clifton*
The MITRE Corporation, Bedford, MA
clifton@mitre.org

Don Marks
Department of Defense
dgm@tycho.ncsc.mil

Abstract

Data mining enables us to discover information we do not expect to find in databases. This can be a security/privacy issue: If we make information available, are we perhaps giving out more than we bargained for? This position paper discusses possible problems and solutions, and outlines ideas for further research in this area.

1 Introduction

Database technology provides a number of advantages. Data mining is one of these; using automated tools to analyze corporate data can help find ways to increase efficiency of an organization.

Another advantage of database technology is information sharing (including sharing with other organizations). For example, publicly accessible corporate telephone books can decrease the need for telephone operators (offloading this task to the caller...) Sharing need not be completely public - making inventory information available to suppliers can help a retail operation to avoid shortages, and can lower the supplier's cost (thus allowing the retailer to negotiate a better price).

These two advantages, when combined, can become a disadvantage. For example, mining a corporate directory to determine staffing of a particular project (and changes in staffing) could help a competitor to determine product rollout dates, allowing preemptive marketing campaigns. Mining retailer inventory data could allow a supplier to determine sales and supplies of competing products, leading to pricing and marketing strategies aimed at reducing the competition (this would be unlikely to be in the best interest of the retailer providing the data).

We will describe this with a more detailed example. Let us suppose that we (as purchasing directors of BigMart, a large supermarket chain) are negotiating a deal with the Dedtrees paper company. They offer to give us a reduced price if we give them access to our database of customer purchases (they say this will allow them to track inventory to allow "just-in-time" production and stocking and reduce their warehouse costs). We accept this deal.

Unbeknownst to us, Dedtrees now starts mining our data. Using a tool for mining sequences [AFS93], they find that customers who purchase cold remedies later purchase facial tissue (allowing them to stock us in advance). They also find (using association rule mining [AIS93]) that people who purchase skim milk also purchase Green paper. Dedtrees now runs a coupon marketing campaign "50 cents off skim milk when you buy Dedtrees products", cutting heavily into the of sales Green paper, who increase prices to us based on the lower sales. When we next go to negotiate with Dedtrees, we find that with reduced competition, they are unwilling to offer us as low a price, and we start to lose business to our competitors (who were able to negotiate a better deal with Green paper).

What are we to do about this? Providing Dedtrees access to our database is good (it improves the efficiency of distribution, lowering costs). Allowing them to mine this data can be good (it helps them to predict inventory needs), but it can also be bad (it allows them to market so as to reduce competition). Some ideas are:

- Limit access to the data. Allow Dedtrees to see only sales of their own products. This prevents

*The work of this author was performed at The MITRE Corporation under support of Department of Defense contract number DAAB07-96-C-E601.

the "good" mining, however.

- Eliminate reference to competing brands. This requires that we determine what the competing brands are, and generate a new view of the data for each supplier.

- Summarize the data. Instead of providing individual purchases, provide only daily summaries. This prevents the mining listed above, but allows general trends, such as "increased sales of cold remedies are followed by increased sales of facial tissue", to be determined.

Alternatively, we may want to prevent mining altogether (such as on a corporate telephone directory). We want people to be able to retrieve individual facts, but we want to protect *any* generalizations that can be formed from mining the data.

In this paper we discuss some specific possibilities; things that can be done to prevent undesirable mining of published data. Section 4 presents directions for research that will support these ends (both direct research on this problem, and work in data mining that can support work on data mining prevention). We will first give an overview of related work that may help to outline this research area.

2 Related Work

The problems of providing access to certain types of information, while restricting access to other types of information (access control) have been studied extensively by the database security community. In particular, multilevel secure databases [ST90] have been developed as a means of enforcing access control policies that limit sharing of information to those with appropriate authority. However, the adequacy of such controls has always been suspect due to the "inference problem": How do we ensure that we cannot infer "private" data from "public" data? Data mining raises the risk factor in such situations, since it allows for the automation of this inference process. There has been progress in this area [Marar], but data mining also poses a new problem: The "inferences" we find are not specific. That is, a rule "A implies B" with confidence 25% may be damaging from a business point of view, but since it is not strictly true it would not appear to be a problem to existing inference techniques.

The secure database community has developed techniques that may help us to control undesired mining problems. An integral part of the data mining procedure is the accumulation of a sufficiently large quantity of information so that generalizations, or rules may be postulated. This "aggregation problem" has been studied most recently in [MMJ94]. Aggregation control techniques can help us to limit access to data, but we are still faced with the problem of determining when an aggregate is too large. In other words, even if we know how to control access to the data, we do not know how much access is too much.

One area that *looks* like a similar problem is disclosure control in statistical databases: Allowing users to gather general statistics on data, while preventing access to individual items (e.g., preventing access to data on individuals from queries over census data [oSM94]). Our concern is exactly the opposite: We want to allow access to the individuals, but prevent generalizations on the whole. However, disclosure control provides a good model of what we would like to see: A strong notion of "how much is too much" when allowing statistical queries.

3 Possible Solutions

We do not believe this is an unsolvable problem. There are a number of things we can do to prevent data mining. We divide these into:

- Limiting Access: If we control access to the data to prevent users from obtaining a sufficiently large and varied sample of the database, we can lower the confidence in the results of any mining that is attempted (does the mined "fact" represent the database as a whole, or is it just an artifact of the small sample?) This is the approach taken by the secure DBMS community.

- "Fuzz" the data: If we alter the data, for example by forcing aggregation into daily records instead of individual transactions or slightly altering data values, we may prevent useful mining while still enabling the planned use of the data. This is an approach used by the U.S. Census Bureau.

- Eliminate unnecessary groupings: Often data values contain unneeded information. For example, U.S. social security numbers are assigned by office (the first three digits identify the office where the social security number was obtained). In addition, the offices often assign these sequentially. Therefore, by grouping social security numbers by the high-order digits, we can group people by location (or location and age) with reasonable reliability. This can provide additional information for use in data mining.

  This can be useful even if we do not *know* how social security numbers are assigned. Simply clustering along the high-order bits of a "unique identifier" is likely to group similar data elements. This similarity may be unknown. It may be chronological if the identifiers are assigned sequentially, or determine the source of the data element if individual data sources are given a "batch" of identifiers. The problem is that this grouping allows us to find similarities that would not be available otherwise (e.g. a high cancer rate among people with similar social security numbers), that lead us to look for further information (what else those people have in common?)

  This can have security implications. For example, an organization that assigns telephone numbers sequentially based on location within a building could find its "phone book" mined to find out who is working on the same projects. Knowledge of the identifier assignment process is not necessary; simply finding a rule that a given group of people working on a *known* project can be determined from grouping on their telephone numbers can lead to the realization that people working on *unknown* projects can be guessed by grouping the telephone number.

  The solution is to ensure that unique identifiers are assigned randomly; thus serving *only* as unique identifiers. This prevents meaningful grouping based on these identifiers, yet does not detract from their intended purpose.

- Augment the data: Sometimes we can add to the data without altering its usefulness. If we know precisely how the data should be used, we may be able to add misleading data that will only be retrieved by inappropriate queries. For example, suppose that a phone book was populated with extra, fictitious, people. Asking for an individual's phone number would return the correct information, but queries to find all individuals in a department would return additional people not even in the company. This requires that we augment the data in non-obvious ways (otherwise it would be simple to reconstruct the original database).

- Audit: While the previous methods address the problems of releasing data "to the world", auditing can be a very effective deterrent against misuse by legitimate users inside an organization. Auditing does not enforce controls, but it may detect misuse so that administrative or criminal disciplinary action may be initiated. Of course, the question here is what should be saved, how can data mining be identified, and what inferences have been formed? Should the audit trail itself be mined in order to answer these questions?

The difficulty with all of these is knowing *when* you have developed a "public" version of the database that is not amenable to mining. To do this requires an understanding of the mining algorithms:

- How do they decide if a given rule or output is "interesting"? Knowing this allows us to alter or limit the data so as to prevent "interesting" rules (or plant false "interesting" rules to mislead the miner).

- What are the performance characteristics? Some mining algorithms experience exponential run times under certain conditions. For example, one algorithm for mining association rules [AIS93] requires an exponential number of passes over the data if the number of items found in each "transaction" is large. We can utilize this to ensure that we make our data computationally infeasible to mine.

This is just a start. Further work is needed to determine just what we can do to prevent unwanted data mining.

4 Further Work

We see a number of areas where work can progress in this area.

4.1 Study of limitations of particular algorithms

Knowing the interest measures of particular algorithms will allow us to protect against them, either by ensuring that the data we publish does not have "interesting" rules, or by providing data that gives "false" rules that hide the real ones. A standard specification of these rules would allow us to develop standard techniques for preventing mining.

Publishing details of algorithms, particularly how interest levels are determined and low-interest items are pruned, will enable countermeasures to those algorithms. We hope we will be able to find commonalities among algorithms, enabling us to develop strategies to defeat a wide variety of data mining tools.

4.2 Definition of "published data"

Another problem is making sure that we *do* make available what we want to. We need to be able to specify what information users are allowed to obtain from the published data. We must, in effect, specify how correct the published view of the data must be. This can draw on work in replicated and mobile databases. That is, how current must a copy be? How correct must a copy be? A specification of the amount of error allowed in data (as done using *quasi-copies* [ABGM90], for example) will allow us to introduce selective error that will limit the effectiveness of data mining.

The issues are different – we are not concerned with cost measures for updating the copy, for example – but some of the same specification mechanisms can apply. Percentage error bounds could be useful. Update counts ("how many updates are allowed before my copy is updated") may not be as appropriate, but careful selection of the "out of date" data may be used to prevent mining.

4.3 Theory of data mining

Developing theories behind different types of data mining (for example, lower bounds on complexity of mining sequences) would enable us to determine if our data is mineable. We could use this to develop "profiles" of hard-to-mine data (for example, a characterization of data that is likely to contain association rules). This will enable us to build systems that provide the data (or access to the data) in a way that ensures that mining is either unlikely to find interesting results, or will find "planted" results that do not concern us.

4.4 Data mining to detect inference problems

Given a database containing both "public" information and "sensitive" information, we can use data mining to search for inference paths from the public to the sensitive information. In other words, a rule with antecedents in the public information and consequents in the sensitive information points to a potential problem. Clustering the public information could conceivably bring together the antecedents. This by itself does not pose a problem, but knowing sensitive information about *one* of the instances could then be conjectured (correctly) to apply to all of them (greatly expanding the "leak" of sensitive information).

Proper use of data mining technology can be used to detect such situations. We can then apply techniques from the previous section to ensure that the data *cannot* be clustered in meaningful ways.

Our introductory example of mining supermarket sales shows how this may work. We may decide to protect brand information; only releasing sales by product type. However, Dedtrees may know that Green Paper dominates a particular product type, and can use data mining combined with this fact to find "brand affinity" information. Preemptive use of data mining (to deduce brand from the information we intend to release) would show this $producttype \Rightarrow brand$ relationship, allowing us to rething our information release strategy.

5 Conclusions

Data mining technology provides a whole new way of exploiting large databases. In the wrong hands, this can be a problem. However, it is not the tools themselves, but the combination of tools *and data* that pose a problem. The obvious solution is to restrict access to data.

Restricting access to data is a tradeoff, however. One of the advantages of database and network technology is the ability to share data, improving

the efficiency of inter-organization operations. We do not want the *threat* of data mining "by the wrong people" to eliminate these efficiencies.

We believe that there is an intermediate solution. We can have the efficiency gains of shared data without "giving away the store". This requires carefully controlling access to the data (or controlling the quality of the data) so as to provide the desired sharing without providing the ability to mine the data.

We have presented some ideas for how to control access, as well as areas for further work. We will continue to work in this area, and hope that others will find this an interesting problem as well.

References

[ABGM90] Rafael Alonso, Daniel Barbará, and Hector Garcia-Molina. Data caching issues in an information retrieval system. *ACM Transactions on Database Systems*, 15(3):359–384, September 1990.

[AFS93] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In David B. Lomet, editor, *4th International Conference on Foundations of Data Organization and Algorithms*, volume 730 of *Lecture Notes in Computer Science*, pages 69–84, Chicago, IL, October 13-15 1993. Springer-Verlag.

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, DC, May 26-28 1993. ACM.

[Marar] Donald G. Marks. Inference in MLS systems. *IEEE Transactions on Knowledge and Data Engineering*, to appear.

[MMJ94] Amihai Motro, Donald G. Marks, and Sushil Jajodia. Aggregation in relational databases: Controlled disclosure of sensitive information. In *Proceedings of the European Symposium On Research In Computer Security*, Brighton, UK, November 1994.

[oSM94] Federal Committee on Statistical Methodology. Report on statistical disclosure limitation methodology. Technical Report PB94-165305, NTIS Document Sales, 1994.

[ST90] Paul D. Stachour and Bhavani M. Thuraisingham. Design of LDV: A multilevel secure relational database management system. *IEEE Transactions on Knowledge and Data Engineering*, 2(2):190–209, 1990.